Lecture 17:

Binary Search Trees; x = change(x)

08/03/22

Upcoming

- A6 due Thursday 8/4 @ 11:59pm
- Checkpoint 7 due Sunday 8/7 @ 11:59pm

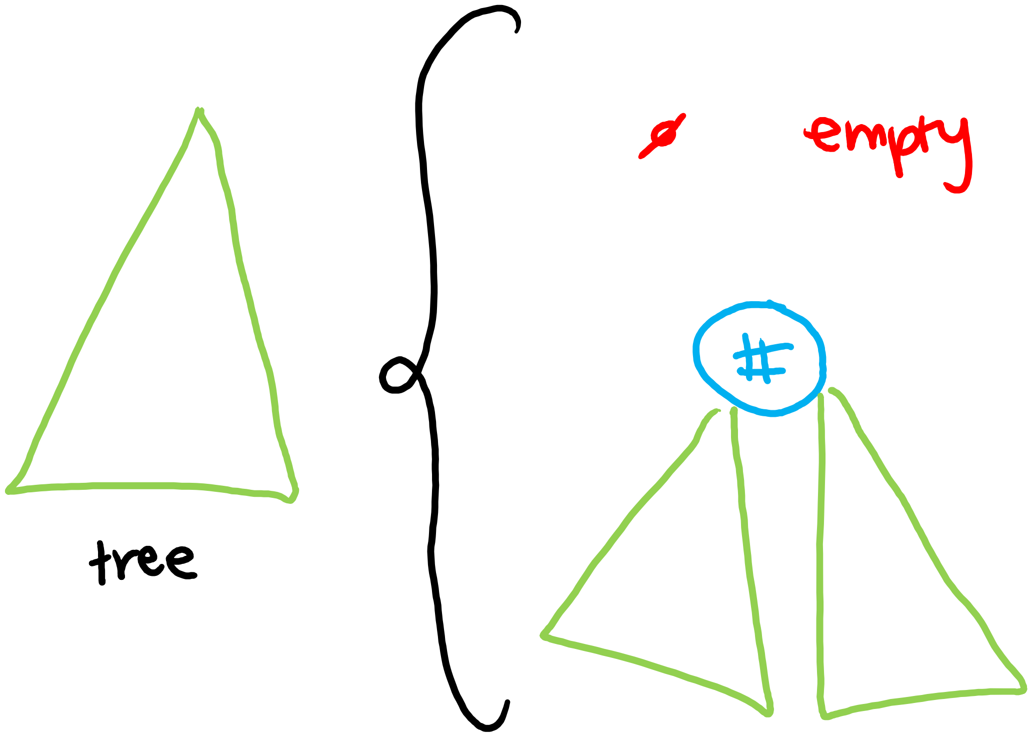
- A7 released today, due Thursday 8/11 @ 11:59pm

- *(A8 will be released Monday 8/8, due Tuesday 8/16)*

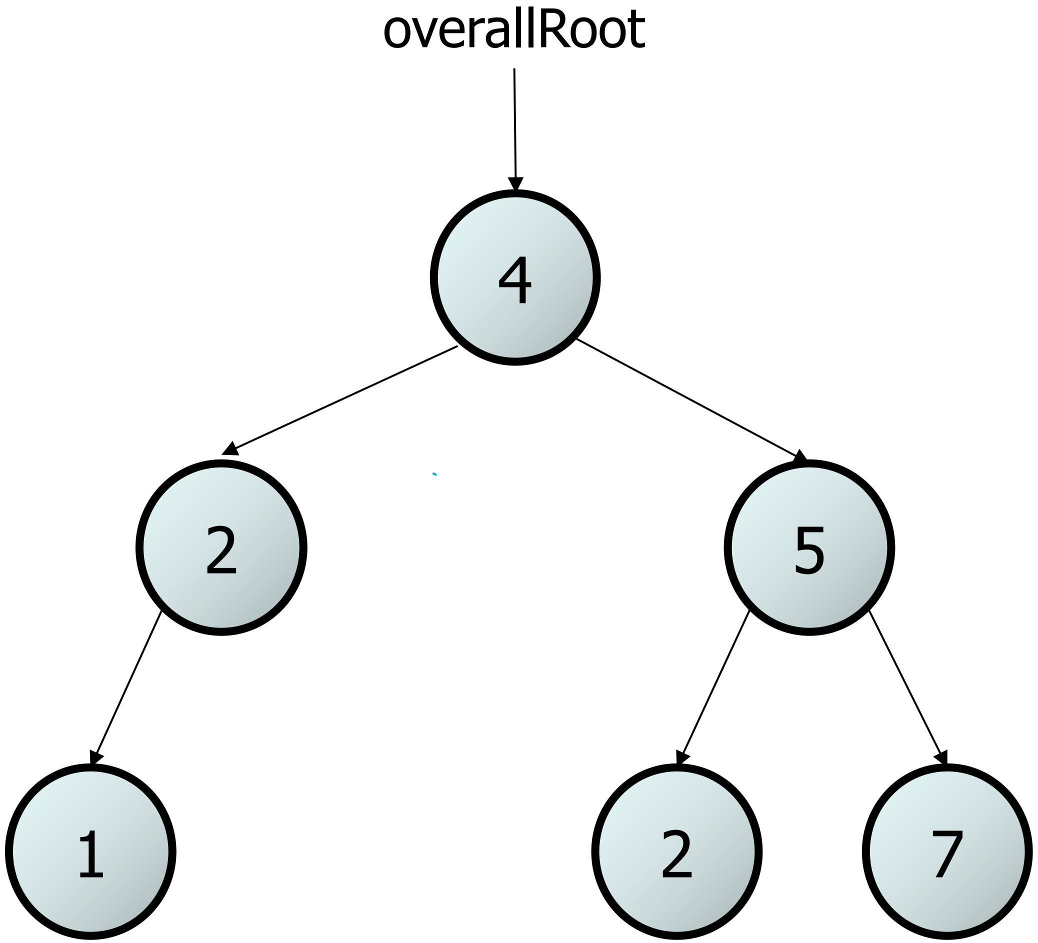
(Recursive) Definition of a Tree

A tree is either:

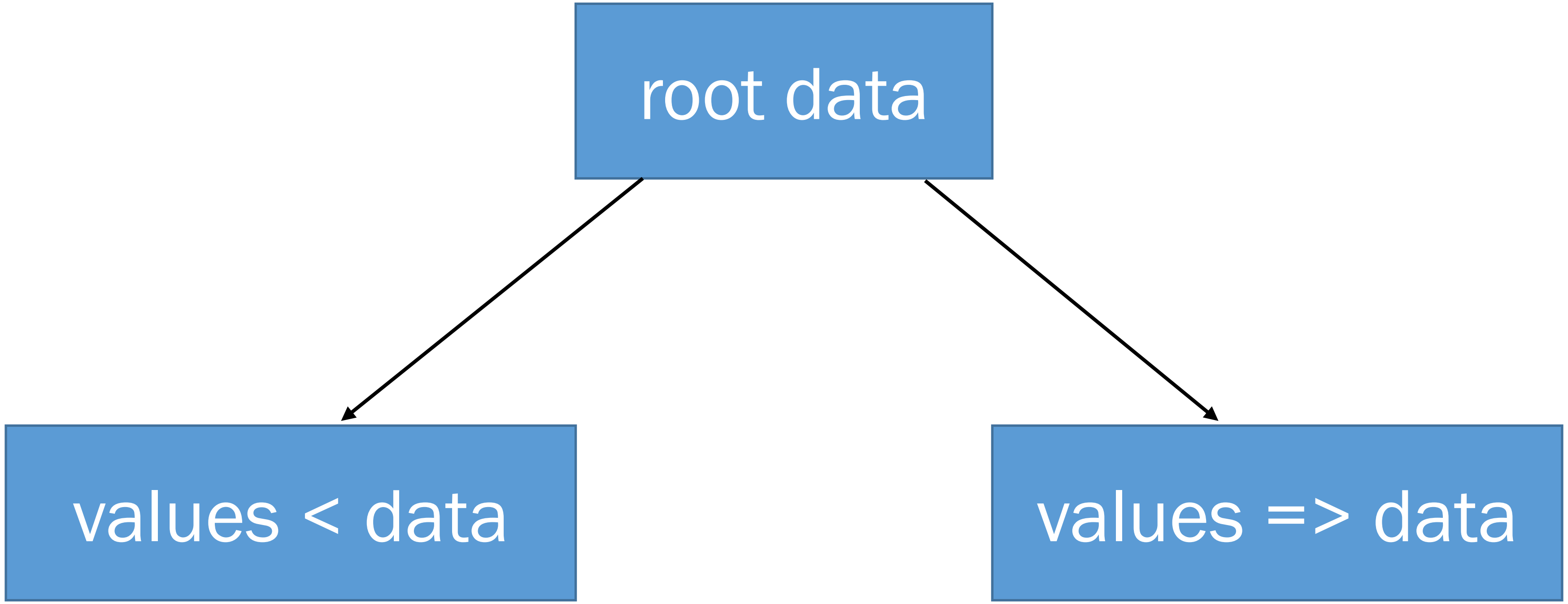
- An empty tree (null), *or*
- A root node with:
  - Data
  - A left subtree
  - A right subtree



Binary Tree



Binary Search Tree Property

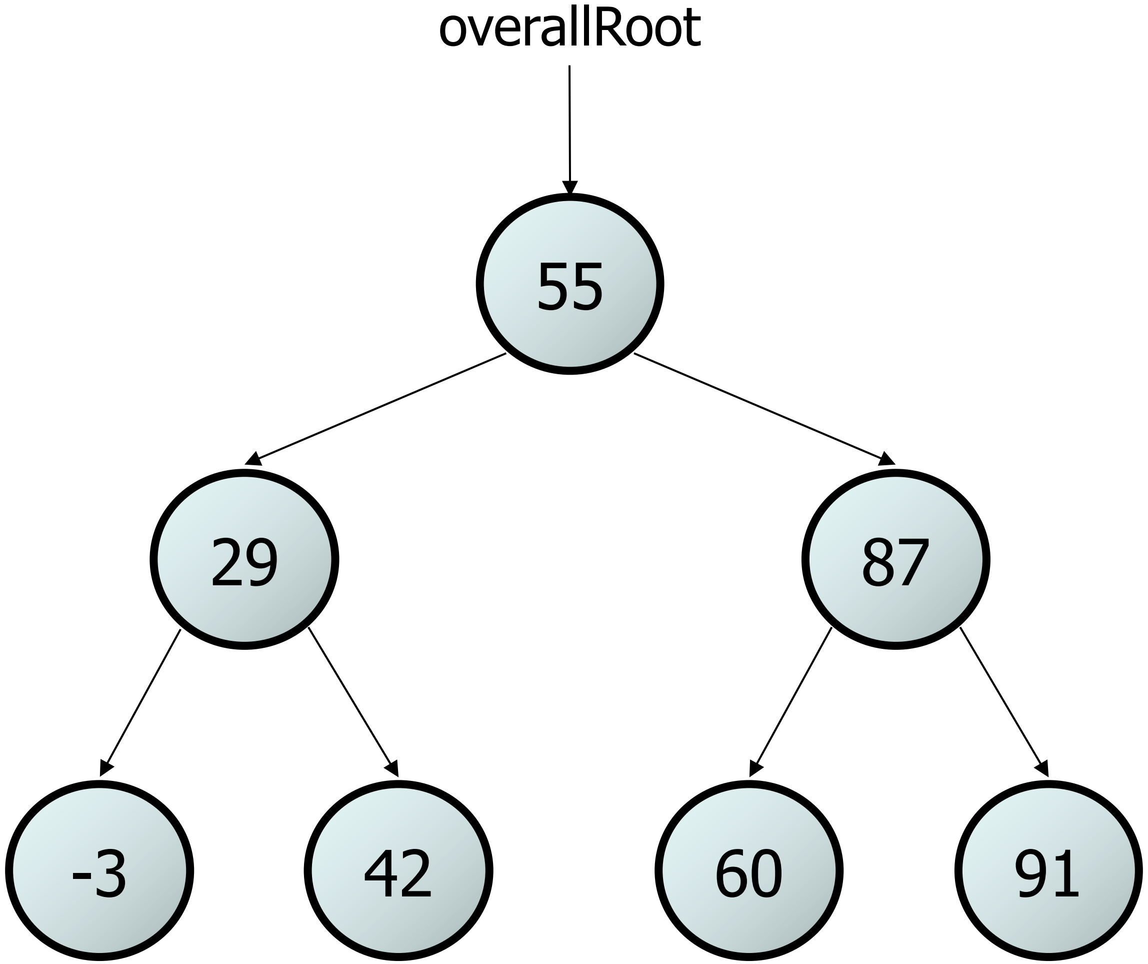


root data

values < data

values => data

Binary Search Tree Example



Point class

```
public class Point {
   private int x;
   private int y;

   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }

   public void setX(int x) {
      this.x = x;
   }

   public void setY(int y) {
      this.y = y;
   }

   public String toString() {
      return "(" + x + ", " + y + ")";
   }

}
```

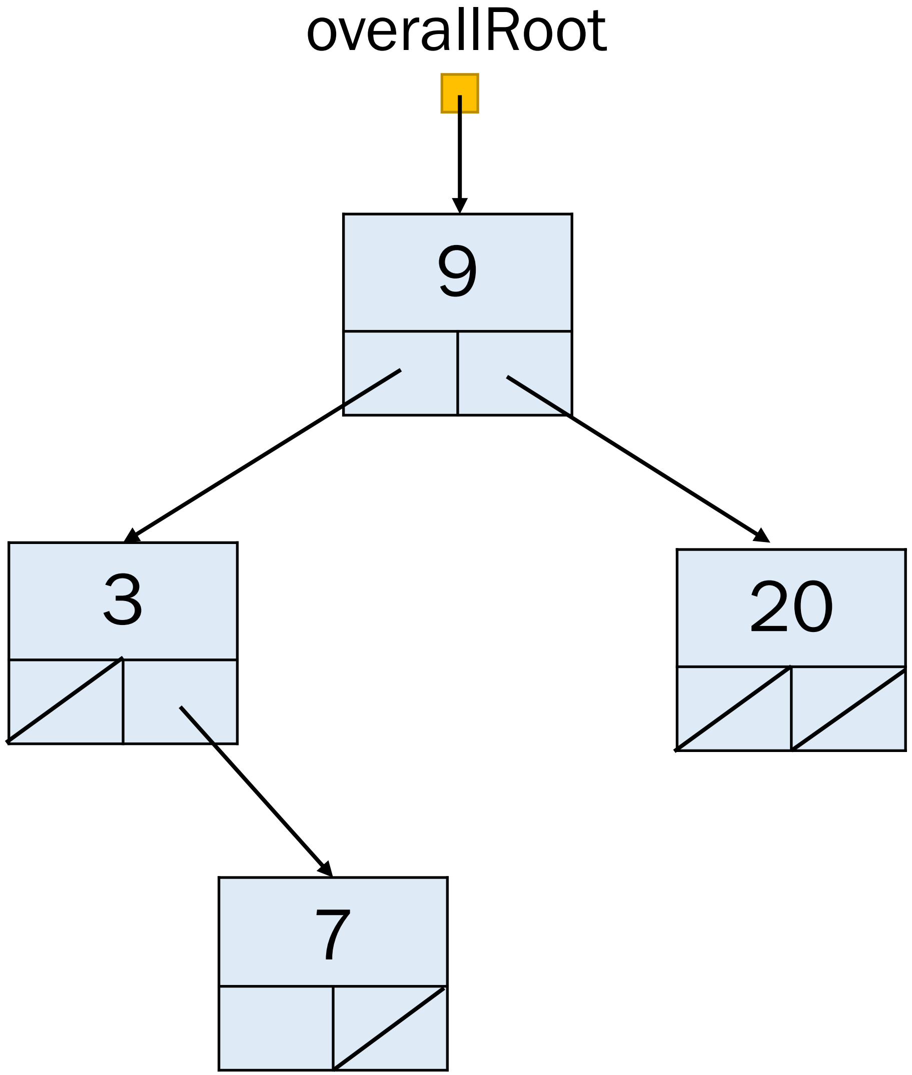
Poll Everywhere

pollev.com/cse143

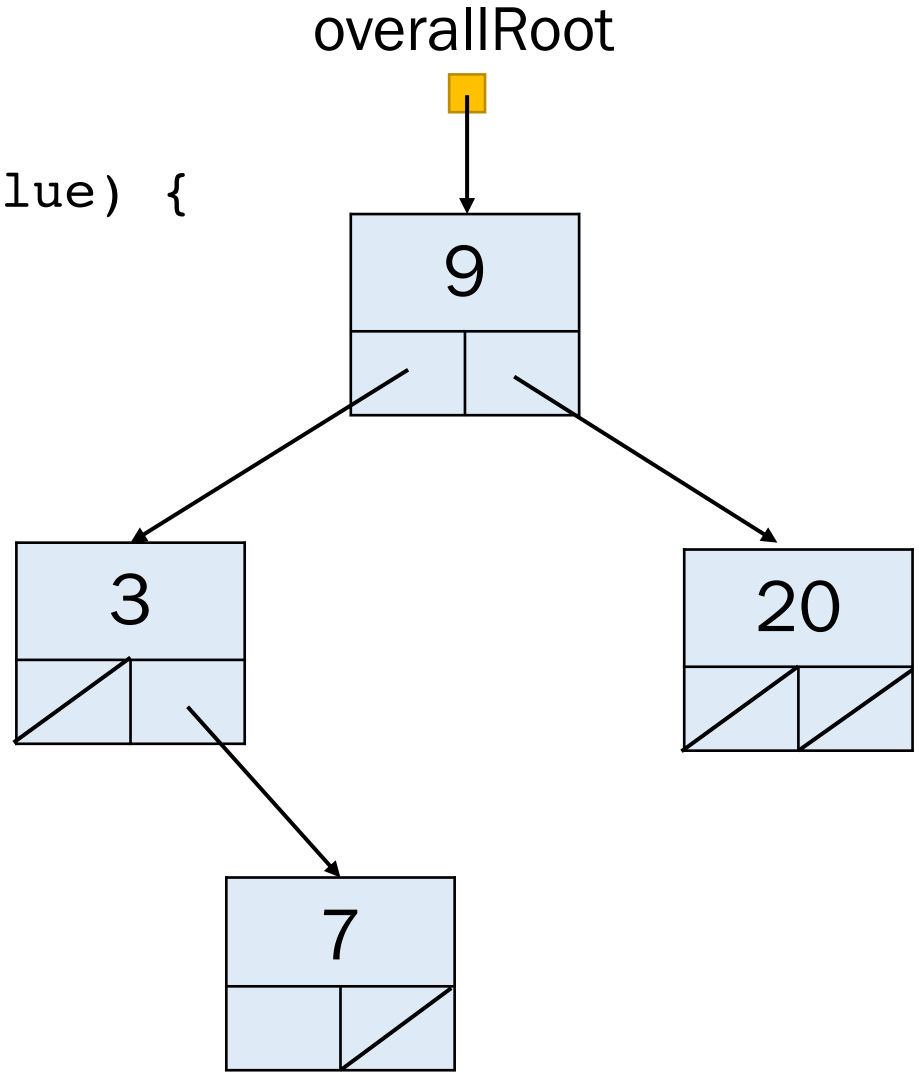
```
public static void main(String[] args) {
    Point p = new Point(1, 2);
    change1(p);
    System.out.println(p);
    change2(p);
    System.out.println(p);
}

public static void change1(Point p) {
    p.setX(14);
}

public static void change2(Point p) {
    p = new Point(7, 8);
}
```

What is the output of this code?

```
public void add(int value) {
   add(overallRoot, value);
}

private void add(IntTreeNode root, int value) {
   if (root == null) {
      root = new IntTreeNode(value);
   } else if (value < root.data) {
      add(root.left, value);
   } else {
      add(root.right, value);
   }
}
```



```
public void add(int value) {
   overallRoot = add(overallRoot, value);
}

private IntTreeNode add(IntTreeNode root, int value) {
   if (root == null) {
      root = new IntTreeNode(value);
   } else if (value < root.data) {
      root.left = add(root.left, value);
   } else {
      root.right = add(root.right, value);
   }
   return root;
}
```


overallRoot
9
3
20
7